Deep learning based super resolution of medical images

Author: Marijn van der Meer
Professor: Pascal Frossard, Supervisor: Mireille El Gheche
Signal Processing Laboratory 4, EPFL Lausanne, Switzerland

***Abstract*—A comprehensive and extensive knowledge of the immunological landscape in tumour stroma can give critical information for the choice of type of immunotherapy and thus improve cancer patient survival. For melanoma, this information is mostly gathered from stroma scans. Frequently, however, stroma scans consist of low resolution image series with a few manually taken high resolution images. An important increase of information content could be obtained if all scans would have high resolution. The goal of this project was to improve low resolution melanoma scans to high resolutions, by a machine learning super resolution model. In order to test the performance of such model, we first produced artificial low resolution images by 2- and 4-fold downsampling and compression from a set of high resolution melonoma snapshots obtained from the CHUV hospital. The resulting low resolution versions were enhanced in a trained model, based on a U-Net architecture with a ResNet encoder and feature loss. This model up-scales low resolution images, taking into account their underlying texture and filling in the unknown details. The resulting high resolution versions were visually close to the ground truth medical images with 2-fold but not 4-fold down-sampled low resolution images. Image comparison measures were largely comparable between high resolution predicted and ground truth medical images, but less accurate for the 4-fold downsampled images. Our results show a very promising development of low to high resolution training. We suspect that results can be further improved by changing the model network's architecture pretrained composites to be trained on data features closer to those observed on medical images.**

I. Introduction

In *"Spatial computation of intratumoral T cells correlates with survival of patients with pancreatic cancer"* [1], Carstens, J. L. et al. correlate spatial placement of intra-tumoral T cells, especially the distribution of cytotoxic T cells near cancerous cells, with the survival of patients with pancreatic cancer. The authors state that *a comprehensive evaluation of the composition and distribution of desmoplastic elements may improve our understanding of how specific stromal composition could impact T-cell activity, with potential impact on the optimisation of immune-modulatory therapies* [1]. This idea is similarly explored for skin cancer in [2], where Bottomley et al. discuss the complex role of the immune system in melanoma stroma, and propose how a deeper understanding of its cellular composition can be channelled into novel therapies. For this, one will need a good evaluation of the composition and distribution of elements active in fibrous and connective tissue growth, and of immune cells in tumour stroma. Ideally, one could use medical stroma images, but as a result of different constraints, these images cannot always provide this information in a straightforward manner. For this project, we have at our disposition melanoma scans provided by the CHUV hospital in Lausanne. These scans consist of slide images of relatively low resolution at high speed, causing an important loss of details but at the upside of having a larger tissue overview. Resolution can be improved manually at desired or preset locations, by zooming in (c.f. Fig. 1). Such high-resolution snapshots come with additional information, namely tissue and cell segmentation, and phenotype unmixing. The unmixing is provided by inForm, a semi-supervised advanced cell analysis software for accurately quantifying biomarkers in tissue sections. Because the number of high-resolution scans is limited, it is difficult to extrapolate from a smaller section to the complete scan, but this would be important in order to properly evaluate the immunological landscape of the whole slide.

The goal of this project is to learn scaling or enhancement of details in an image to improve resolution. What we would like to achieve is to understand how a low resolution image can be scaled up to a higher resolution where the unknown details in the output are filled in. We aim to develop a model that is able to produce high resolution images from low-resolution snapshots by taking into account the high resolution snapshots and the underlying image structure. In the future this may help to produce high-resolution images of whole melanoma

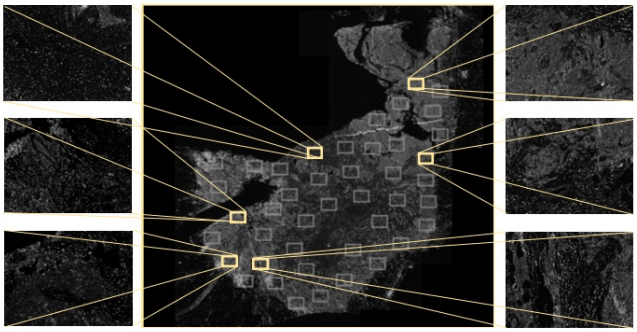


Fig. 1: Example of a low resolution melanoma stroma image with six high-resolution snapshots. Left and Right: example of high-resolution snapshots. Middle: low-resolution of the whole slide with 45 snapshot positions (small boxes).

landscapes from low resolution scans.

II. State of the art

A general way to re-sample a low-resolution image to high resolution is to apply some form of pixel interpolation by bicubic, bilinear, or nearest neighbour methods. However, scaling techniques such as interpolation and nearest neighbour miss fine details, can cause overshooting, clipping and may lead to blurring. The problem that super resolution based on machine learning tries to overcome is to interpolate from low to high resolution without such biases. A variety of elaborate deep-learning models are being developed, which base on training from available high-resolution images the rules for low to high interpolations.

Most current super-resolution models based on deep learning are trained by means of Generator Adversarial Networks (GANs) [3]. In [4], Hongda Wang et al. present a deep-learning-enabled super-resolution model across different fluorescence microscopy modalities. Their method is based on training a generative adversarial network to create high resolution total internal reflection fluorescence (TIRF) microscopy images of sub-cellular structures from diffraction-limited input images. Their method showed good performance. However, one limitation of GANs is that their loss function is formed as part of the process and is not explicitly tailored for the task of super resolution. Furthermore, GANs have a tendency to be unstable and take a long time to converge. Therefore, this type of structure was not preferred for this project.

In contrast, Zhang et al. [5] suggest a method for fast single image super-resolution (FMISR) with three hidden layers in addition to a sub-pixel convolution layer and a mini-network. They argue that their model improves the speed and quality of image reconstruction. This model was used and trained on MRI and CT scans of the retinal macula, brain and bones, and thus focused on a broad overview of those structures without cellular composition. As such, we believed that it would not be ideal for melanoma stroma scans, which are more focused on cell types.

Instead, we base our model on the deep learning architecture presented by Jeremy Howard and Rachel Thomas in the Fastai course [6] as it promises satisfactory performance, producing even better high resolution images than the original in some cases on ImageNet. In their course, Howard et al. first performed super resolution with GANs on ImageNet but were unsatisfied with the results. In an attempt to get rid of them completely and avoid their long training time, they produced a deep learning model that was based on a loss function specifically designed for super-resolution (c.f. section III-B). We chose to adapt their model to our medical images, as it seems to come with high performance, relatively short training times and precise documentation.

III. Models and Methods

A. Data pre-processing

1) HR data: The data provided by the CHUV hospital consisted of 5-channel low-resolution raw *im3* files and approximately thirty 35-channel high-resolution snapshots per low-resolution scan. Additionally, we had inForm outputs for each high-resolution patch concerning its location, 3-channel phenotype images (c.f. Fig. 2), cell labelling data, and cell and tissue segmentation images. The phenotypes detected by inForm are the following:

- DAPI: blue-fluorescent DNA stain used for cell-nucleus staining in fluorescence microscopy.
- CD4: glycoprotein found on the surface of immune cells such as T lymphocytes. As mentioned in [1] and [2], knowledge of the distribution of immune cells in proximity to cancerous cells can give crucial information about patient's survival and may influence the choice of therapy.
- CD3: protein complex and T lymphocyte cell co-receptor.
- CD8: transmembrane glycoprotein that serves as a co-receptor for the T-cell receptor.
- FOXP3: protein involved in regulating the development and function of regulatory T lymphocyte cells.
- CK: cytokeratin, intermediate filaments forming the proteins that provide mechanical support for the

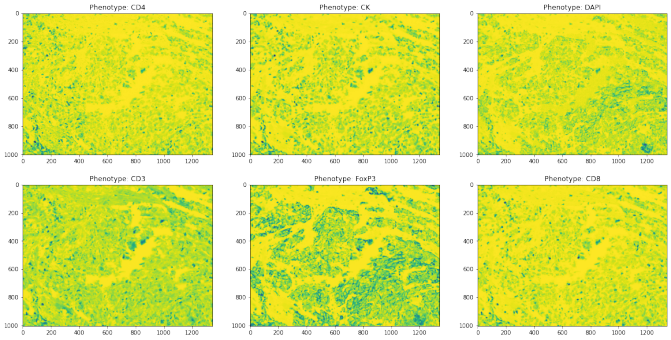


Fig. 2: Example of phenotype image provided by inForm. 3-channel images of size (3, 1004, 1334). From top left to bottom right: CD4, CK, DAPI, CD3, FoxP3 and CD8.

cells. CKs are important markers for tumour cells. Primarily, metastases of a given carcinoma share a common cytokeratin pattern, which distinguishes them from other types of cancer, thus making it possible to distinguish between different tumours.

This project started with a simple approach, with the idea to be able to add more complexity downstream where necessary. Our first objective was to find a model capable of predicting high-resolution images from low-resolution inputs assuming high-resolution data were supplied. Consequently, only high-resolution phenotype files were relevant.

As a result, two types of data sets were created from the high-resolution images. The first data set, used as input in the primary stage of the development of the model, was comprised of 3 channel images of size (3, 1004, 1344). Each image was created by transforming the phenotype files from *tiff* format to *JPEG*. This was necessary because the model accepted only this format of images at that stage. Once this was corrected, a second data set was produced by creating *tiff* files of size (3, 1004, 1344) where each channel was a converted grayscale phenotype image (c.f. Fig. 3 for an illustration of the process and Fig. 4 for an example). So, for a certain location and patient, two files were created. The first one comprised the phenotypes DAPI, CD4 and CK, and the second one CD3, CD8 and FOXP3. The choice of which phenotypes were put together was arbitrary. Finally, because images of size (3, 1004, 1344) required very small batch sizes in order to avoid GPU memory problems, HR images were cut into 4 parts. Therefore, for the rest of this paper, if not otherwise specified, HR images are of size (3, 502, 672).

2) LR data: The corresponding low-resolution snapshots to the high-resolution snapshots could have been recovered from the whole LR scan at the precise location

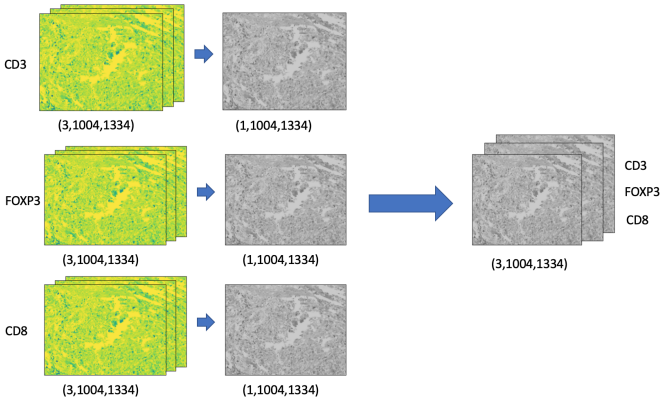


Fig. 3: Process of creating the input data to the model. Three phenotype files of size (3, 1004, 1334) are converted to one grayscale channel and assembled into a new 3 channel file of the same size where each channel is one grayscale phenotype.

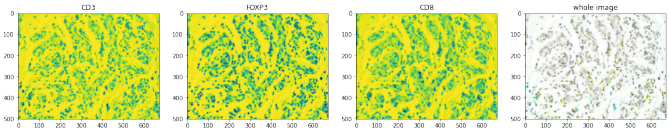


Fig. 4: Example of an (3, 1004, 1334) image created as HR input for the model as seen in 3. CD3 (left), FOXP3 (middle left) and CD8 (middle right) are each one channel from the composite image (right). Channels are coloured.

of HR images. However, out of simplicity, we created the LR data directly from the HR images. This allowed us to spend more time on building the quality of the model and save time from selecting and resizing LR images from original scans. To transform into LR, the HR images (3, 502, 672) were scaled down by bicubic interpolation by a factor of 2 to size (3, 250, 334) or by 4 to (3, 125, 167) and were saved in lower quality (60 in JPEG compression). As the simulations with LR data of size (3, 125, 167) presented the best results, the rest of this paper will concentrate on that size for LR images.

B. Model

The model takes an LR image as input, passes it through a trained neural network and generates an image of the same size that is an enhancement of the input. It is built on the approach introduced in the Fastai course by Jeremy Howard and Rachel Thomas [7] and [6]. It uses the Fastai software library and the PyTorch deep learning framework. Overall, their method is built on the U-Net architecture [8] with cross-connections, a ResNet-34 based encoder and decoder, and a loss function based on activations from a VGG-16 model, pixel loss and gram matrix loss. Furthermore, progressive resizing and discriminative learning rates, or differential learning as

it is sometimes called, are employed to enhance the model's performance.

1) Residual Networks (ResNet): ResNets are Convolutional Neural Network (CNN) architectures made up of series of residual blocks (c.f. Fig. 9 in section VI). ResNet is different from classical CNNs because of its skip connections, designed to address the vanishing gradient problem [9]. Cross-connections between layers in ResNet make it possible to skip large sections if required and create a smoother loss surface. As mentioned above, ResNet-34 is used as an encoder in our U-Net architecture.

2) U-Net: U-Net is a CNN architecture specifically developed for biomedical image segmentation [8]. U-Nets show high performance in segmentation tasks and image processing, such as super resolution. They have been found to be particularly effective in cases where the output and inputs are of similar size. The network is U-shaped (c.f. Fig.5), as it is divided into a down-sampling/encoder section that forms the left side and an up-sampling/decoder for the right.

For the decoding portion of the network, the model adds pixels around and between the existing pixels to achieve the desired higher resolution. In this model, pixel shuffle/sub-pixel convolution with ICNR initialisation is used instead of traditional interpolation methods. According to [6] and [10], this results in a superior quality of pixel gap filling. Pixel shuffling scales up by a factor of two, then further replication padding produces an extra pixel around the image and the average pooling extracts features smoothly. Later convolutions then enhance the details within the image before moving to another step and doubling dimensions [7].

Traditional U-Net architectures result in a lack of fine detail in super resolution. To avoid this, skip connections cross from same sized parts from the encoder to the decoder (grey arrows in Fig. 5). This is done by setting ResNet-34 as an encoder and U-Net constructs the decoder side such as to have a network with cross connections (done automatically in Fastai). ResNet-34 is loaded as pre-trained on ImageNet to speed up training time and to use transfer learning [11].

3) Feature loss: The loss function used in the Fastai course is inspired from the feature loss from [12] but it is combined with a mean absolute error pixel loss and a gram matrix loss. This was found to be more efficient than single feature loss in [6] as well as by the Fastai developers. Feature loss is created from the intermediate activations in the backbone of the VGG-16 network (c.f. Fig. 10 and 11 in section VI). Normally, VGG-16 is used for image classification, however, rather than taking the final output of VGG, an intermediate output is taken. This gives information about the features in the images (e.g. in this part of the image there is something circular, shiny, etc). To compute the loss during training, the prediction and target are put through the pre-trained VGG-16 network and the activation of the same layer for the prediction and target are then compared with L1 loss (MSE can also be used). This creates the feature loss mentioned above which allows the loss function to know which features are in the ground truth image and to evaluate how well the model's predicted features are consistent with them, rather than evaluating only pixel differences. In our case, three feature losses are evaluated, one for each dimension of max-pooling layers in VGG-16. As mentioned above, the model also computes pixel loss (L1 loss between pixels) and gram matrix loss, the L1 loss between the gram matrix of the predicted and target feature. Gram matrix loss is known as style loss because it carries information about an image's style (texture) and not its spatial structure [13]. Note that feature loss relies on VGG-16 being pre-trained. By default, VGG-16 from PyTorch is pretrained on ImageNet and the same applies to ResNet-34. Therefore our images have to be normalised to ImageNet statistics, e.g. mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225].

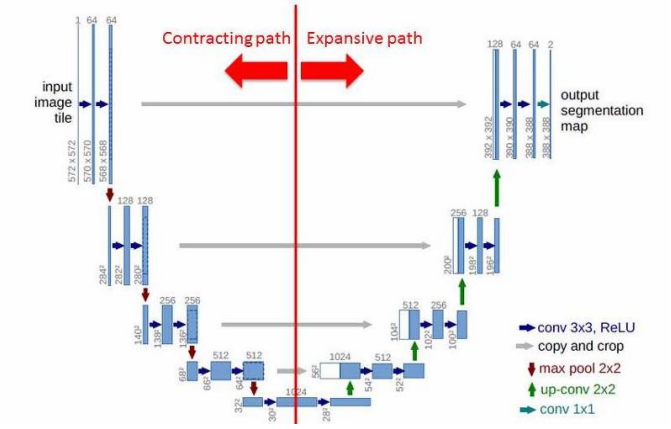
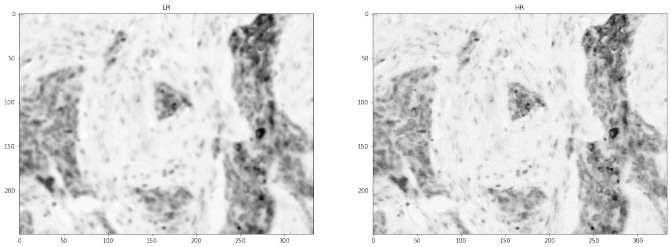


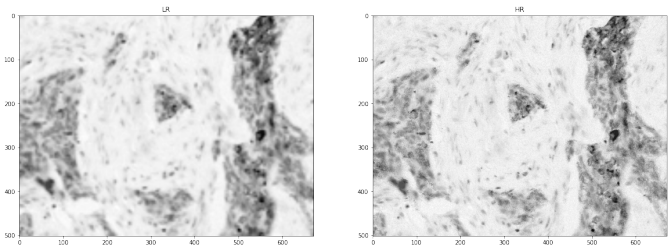
Fig. 5: U-Net architecture with cross connections (example for 32x32 pixels in the lowest resolution) [8]. Cross-connections are visible in grey. Encoder/down-sampler part is on the left and decoder/up-sampler on the right.

C. Training

Training is carried out on the model mentioned above; a U-Net based on a ResNet-34 architecture pre-trained on ImageNet and applying a feature loss in combination with a pixel and gram matrix loss. The feature



(a)



(b)

Fig. 6: Example of training data for phases 1 and 2. a) Phase 1. Left: low resolution images up-sampled from (3, 125, 167) to (3, 250,334). Right: high resolution down-sampled from (3, 502, 672) to (3, 250, 334). b) Training data for phase 2. Left: low resolution images up-sampled from (3, 125, 167) to (3, 502, 672). Right: high resolution of same size.

loss function is based on the VGG-16 architecture (c.f. section III-B3), itself pre-trained on ImageNet .

1) Progressive resizing: The training of this model was conducted in two phases. The first phase (phase 1a and phase 1b), as shown in Fig.6a, took low resolution images of size (3, 250, 334) as input and produced output high resolution images of the same size. To prepare the data for this phase, high and low resolution images were subsequently down/up-sampled using bicubic transformation by a factor of approximately 2. For the second phase (phase 2a and 2b), as shown in Fig.6b, low resolution images were scaled up by a factor 4, giving an input and output of size (3, 502, 672). Moreover, in addition to the resizing, the images for both phases also underwent several other transformations to prevent the model to overfit and encourage generalisation by learning how to improve all kinds of degraded image shapes. These transformations include [7]:

- Random quality reduction
- Random crops
- Flipping the images horizontally
- Lighting adjustment
- Perspective warping
- Random noise

This approach of training on smaller images first and, once the loss decreased enough, transfer the model's knowledge to a new model that accepts bigger resized images for the second phase, is called progressive resizing [14]. Because of the range of diversity in images the model sees this way, it makes it less susceptible to overfitting. Note that the number of weights does not change between phases.

2) Freezing gradient descent: As mentioned above, U-Net is split into an encoder (left) based on ResNet-34 and a decoder (right). Because the encoder comes with pretrained weights but the decoder is randomly initialised, the encoder first needs to be frozen (phase 1a and 2a) so that only the weights in the decoder are trained initially. Then, once the decoder is trained, the encoder is unfrozen and the model resumes training some more to be able to take advantage of the encoder's pretrained knowledge (phase 1b and 2b).

3) Discriminative learning rates: Once the encoder is unfrozen, the model is trained with discriminative learning rates [15] (phase 1b and 2b), a method where different learning rates are set to different layers in the network during training. The intuition behind this is that the first few layers will typically contain very granular details (e.g. lines and edges) and is thus information that should be retained [15]. Therefore, there is not much need to change their weights by a big amount. On the other hand, detailed features appearing in later layers might not necessarily need to be kept. In our case, learning rates are much smaller in the encoder ($1e^{-6}$ to $1e^{-5}$) than in the decoder ($1e^{-4}$ to $1e^{-3}$). This allows for fine-tuning once the encoder is unfrozen, without risking losing its already acquired accuracy.

For further details of other parameters used, see Tables II, III, IV and V.

IV. Results

A. Testing data

In order to evaluate the predictions of the model trained as described in section III-C, two data sets were created. High-resolution test images were scaled down to medium (MR) and low-resolution (LR) images of size (3, 250, 334) and (3, 125, 167), respectively, by bicubic interpolation, and compressed to a lower quality of 60 in JPEG compression. Both test sets were then up-scaled to size (3, 502, 672) by bicubic interpolation as this was the input size of the last training phase (c.f. Fig. 7). Note that, although they were of the same size as HR, they were of lower resolution as they had been

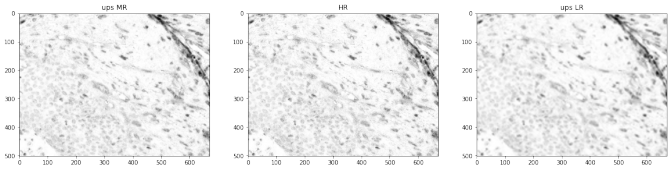


Fig. 7: Example of HR images created by bicubic interpolation from LR (right) and MR (left) compared to the ground truth HR (middle). All images are of size (3, 502, 672).

compressed. These images were also used as a baseline for the predictions, the goal being to do better than the high resolution images created by interpolation. As is noticeable by eye in Fig. 7, both interpolated images are of lower resolution compared to the ground truth image.

B. Training loss

As shown in Tables II, III, IV and V, a lot of different simulations were run with varying image sizes and parameters. The results of the simulation with the best results (Simulation 9) are presented in this section. Training and validation errors decreased steadily until they stabilised in phase 1a and 2a. However, errors during phase 1b and 2b, once the encoder has been unfrozen, did not decrease as significantly (c.f. Fig. 12 in section VI). This is not alarming as unfreezing the encoder is done in order to fine-tune the parameters of the model. However, it is also noticeable that in phase 2, there might be a slight overfitting of the model to the training data as the validation error is somewhat higher than the training error.

C. Predictions

An example of predictions made by the model can be seen in Fig. 8, and others in larger size in Fig. 15, 16. Visual observation suggested that the images predicted by the model are of higher resolution than the interpolated images. Furthermore, the results from medium resolution test images showed a closer resemblance to the ground truth resolution then from low resolution. In a first attempt to evaluate mathematically whether the predicted images were better than bicubic interpolation, the following traditional metrics were used:

- MAE: L1 pixel loss, used as base loss in the model (c.f. III-B3). Measures similarity between pixels.
- MSE: L2 pixel loss. Measures similarity between pixels.
- NMSE: normalised L2 pixel loss. Measures similarity between pixels.
- SSIM: structural similarity (SSIM) index is a method used for measuring the similarity between two images [0,1] where 1 indicates fully identical images.
- PSNR: peak signal-to-noise ratio, used to evaluate predictions in [5]. The bigger the ratio, the higher the similarity between images.

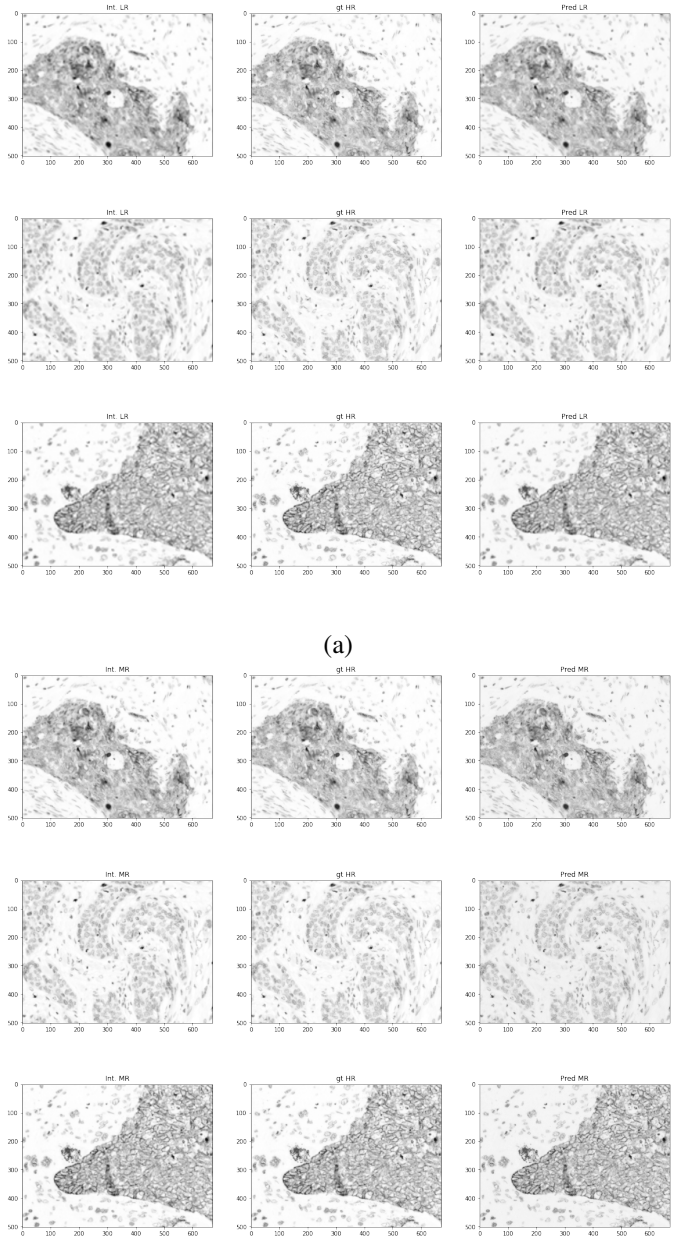
According to the traditional measures (c.f. Fig. 13 in section VI), the interpolated high resolution images were closer to the ground truth for most of the tests. This was surprising as visually it was noticeable that the predictions were in fact of higher resolution. Therefore, a second set of measures were established directly from the losses used in the training of the model (c.f Fig. 14 in section VI):

- pixel loss: L1 loss between pixels (same as MAE from above).
- feature loss: L1 loss between target and predicted features (c.f. section III-B3)
- gram matrix loss: L1 loss between gram matrix of input and output features
- overall loss: sum of all loss measures

With these new evaluations, it was clear that for most of the losses (except for pixel loss), predictions fared better than their interpolated counterparts. Moreover, the difference seemed greater for the predictions from medium resolution test images, which confirmed what was already visible by eye.

V. Discussion

To the naked eye, the model's predictions were of higher resolution than the interpolated images, and they strongly resembled the real high-resolution images. The images predicted by the model seemed to be a shade darker in contrast, but it was not very clear how or where this originated in the process. It is however more challenging to prove mathematically that the predictions were significantly better than bicubic up-sampling. Although the measures presented in Fig. 14 seemed to show that the predictions on most of our tests perform better than the corresponding interpolated image, the difference should ideally be more import. In [7], test were made with this model on photographs of plants, animals and humans. C. Thomas produced results where predictions were sometimes of even higher resolution than the original ground truth. This is clearly not the case for our images. We suspect that this problem arises when using this model on medical images, as a crucial part of its architecture relies on the pretrained ResNet-34 and VGG-16 (see section III-B3). As a reminder, ResNet-34 is used as encoder in the U-Net and VGG-16 used to evaluate the loss but both are pretrained on ImageNet, a



(a)

(b)

Fig. 8: Example of predictions made by the model from LR and MR test images (right) compared with ground truth HR (middle) and bicubic interpolated HR versions (left). For more examples and bigger images, see section VI Fig. 15, 16. a) Prediction and interpolation from LR test images. b) Prediction and interpolation from MR test images.

dataset that mostly includes flora, fauna, natural objects, geological formations and people. Therefore, when looking for features during training to evaluate the feature and gram loss, it might be more difficult for the model to detect features in the used melanoma scans, as the images differ considerably from what it had been trained on. As can be seen in [16], the kind of features typically detected by VGG-16 are contours of faces, eyes, contours of objects, etc. In contrast, melanoma scans show cellular structures with a lot of important but small details, presenting features that are quite different from what the model has been trained on to detect. Ideally VGG-16 and ResNet-34 should have been pretrained on an image set that is closer to our data. But, to support this claim, a feature map similar to what was done in [16] should be produced on our data to evaluate exactly what features are detected in our images. Due to time constraints, it was not possible to evaluate this here.

A second issue seems to be the image contrast, which should be addressed in detail as it might indicate a need for post-processing on the predicted images. Although training images were normalised within ranges of (0,1), the prediction of the model was not in the same range, and the outcome showed pixels with negative values or bigger than 1. To evaluate similarity, a min-max scaler was used in this model, as it was the simplest approach to post-process the predictions in a pixel range of (0,1). However, the pixel value range should be looked at in detail and other normalisation methods should be considered to see whether the contrast can be improved. Again, because of time constraints, it was unfortunately not possible to analyse this.

Finally, in order to, in the future, recreate a full high resolution image of the whole melanoma scan, the model should be retrained on low resolution data from the original scan and not from down-sampled images.

REFERENCES

[1] J. L. Carstens, P. Correa de Sampaio, D. Yang, S. Barua, H. Wang, A. Rao, J. P. Allison, V. S. LeBleu, and R. Kalluri, "Spatial computation of intratumoral t cells correlates with survival of patients with pancreatic cancer," *Nature Communications*, vol. 8, no. 1, p. 15095, Apr 2017. [Online]. Available: https://doi.org/10.1038/ncomms15095

[2] M. Bottomley, J. Thomson, C. Harwood, and I. Leigh, "The role of the immune system in cutaneous squamous cell carcinoma," *International Journal of Molecular Sciences*, vol. 20, p. 2009, 04 2019.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," *ArXiv*, vol. abs/1406.2661, 2014.

[4] H. Wang, Y. Rivenson, Y. Jin, Z. Wei, R. Gao, H. Günaydın, L. A. Bentolila, C. Kural, and A. Ozcan, "Deep-learning enables cross-modality super-resolution in fluorescence microscopy (Conference Presentation)," in *Optical Data Science II*, B. Jalali and K. ichi Kitayama, Eds., vol. 10937, International Society for Optics and Photonics. SPIE, 2019. [Online]. Available: https://doi.org/10.1117/12.2507596

[5] S. Zhang, G. Liang, S. Pan, and L. Zheng, "A fast medical image super resolution method based on deep learning network," *IEEE Access*, vol. 7, pp. 12 319–12 327, 2019.

[6] C. Thomas, "U-nets with resnet encoders and cross connections," 03 2019. [Online]. Available: https://towardsdatascience.com/u-nets-with-resnet-encoders-and-cross-connections-d8ba94125a2c

[7] ——, "Deep learning based super resolution, without using a gan," 02 2019. [Online]. Available: https://towardsdatascience.com/deep-learning-based-super-resolution-without-using-a-gan-11c9bb5b6cd5

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[10] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," *CoRR*, vol. abs/1609.05158, 2016. [Online]. Available: http://arxiv.org/abs/1609.05158

[11] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," *CoRR*, vol. abs/1808.01974, 2018. [Online]. Available: http://arxiv.org/abs/1808.01974

[12] J. Johnson, A. Alahi, and F. Li, "Perceptual losses for real-time style transfer and super-resolution," *CoRR*, vol. abs/1603.08155, 2016. [Online]. Available: http://arxiv.org/abs/1603.08155

[13] R. Asokan, "Neural networks intuitions: 2. dot product, gram matrix and neural style transfer," 01 2019. [Online]. Available: https://towardsdatascience.com/neural-networks-intuitions-2-dot-product-gram-matrix-and-neural-style-transfer-5d39653e7916

[14] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *ArXiv*, vol. abs/1710.10196, 2018.

[15] H. Zulkifli, "Understanding learning rates and how it improves performance in deep learning," 01 2018. [Online]. Available: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10

[16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901

[17] Nov 2018. [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/

VI. Appendix

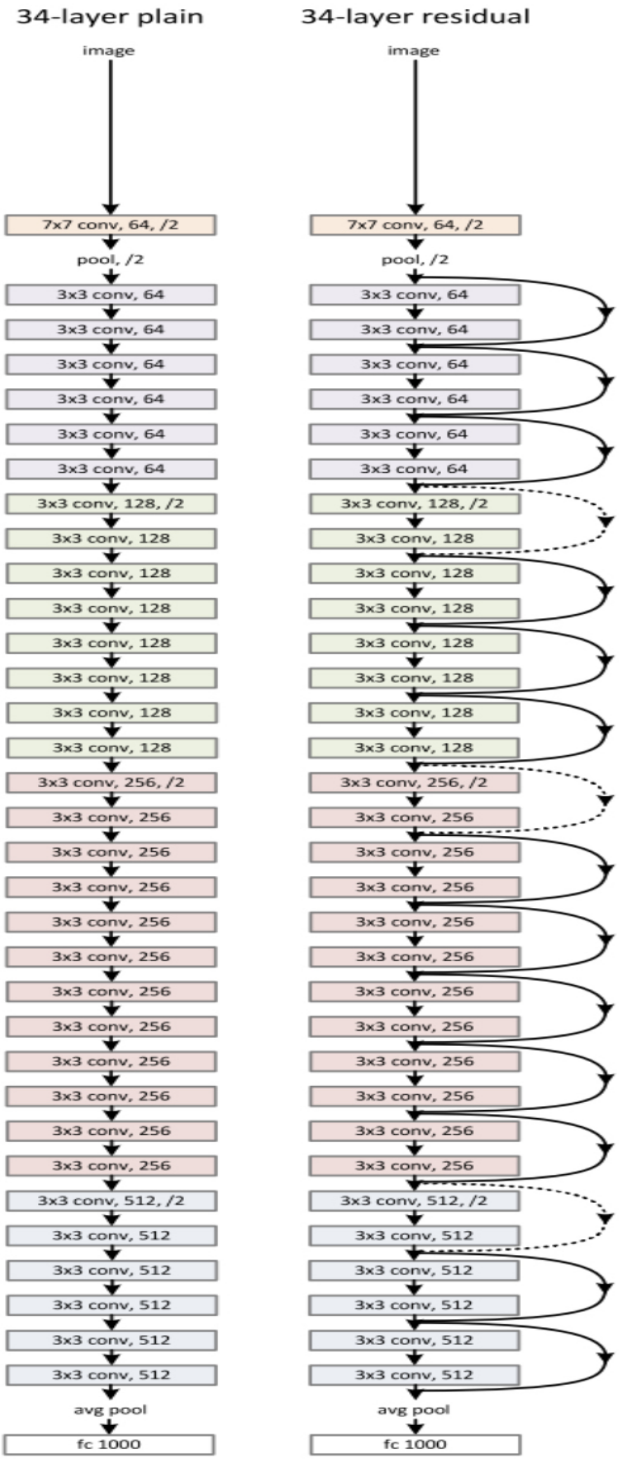


Fig. 9: Left 34 Layer CNN, right 34 Layer ResNet CNN [9].

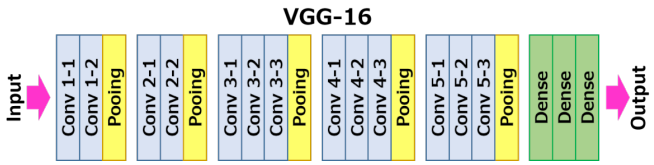


Fig. 10: Different layers in VGG-16 [17].

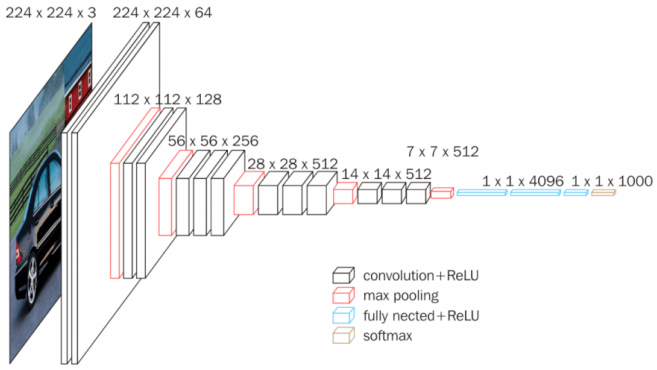
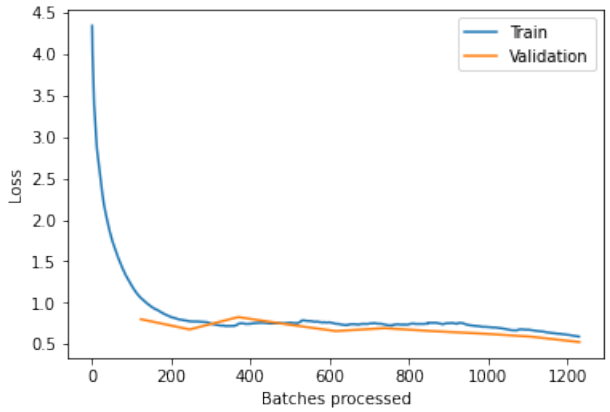
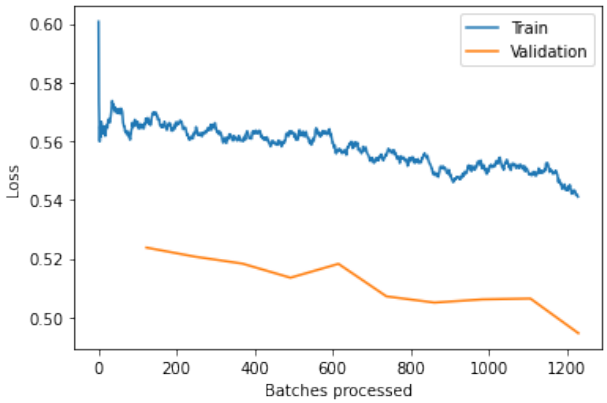


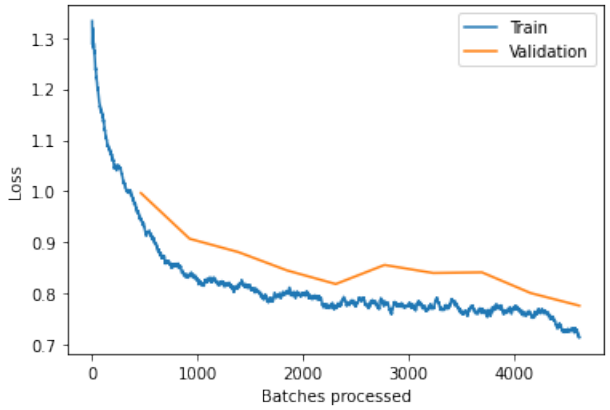
Fig. 11: VGG-16 architecture [17].



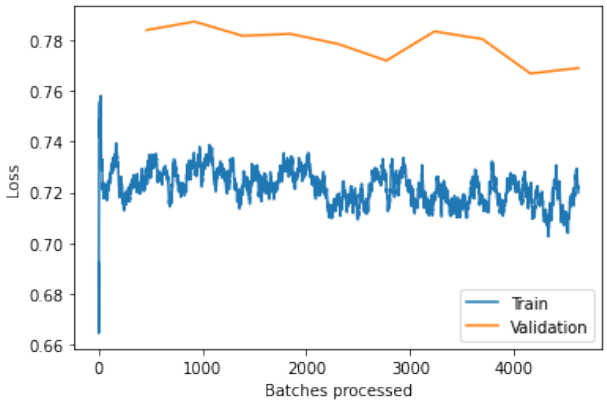
(a) Phase 1a



(b) Phase 1b

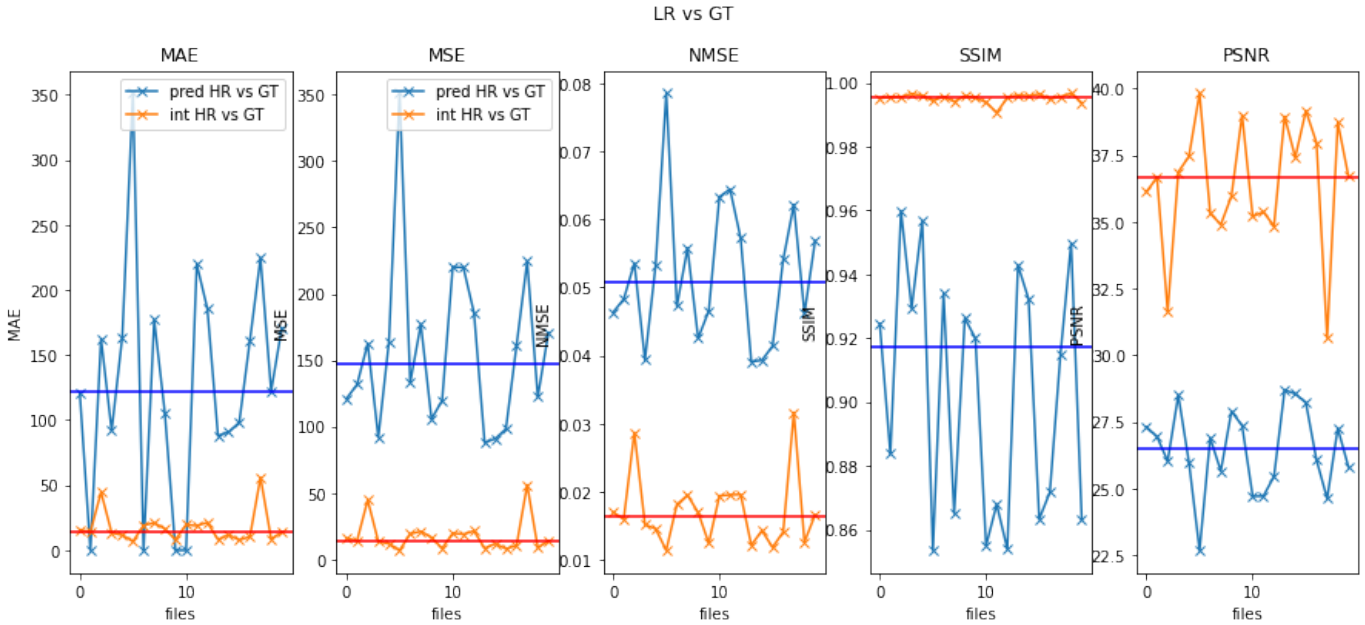


(c) Phase 2a

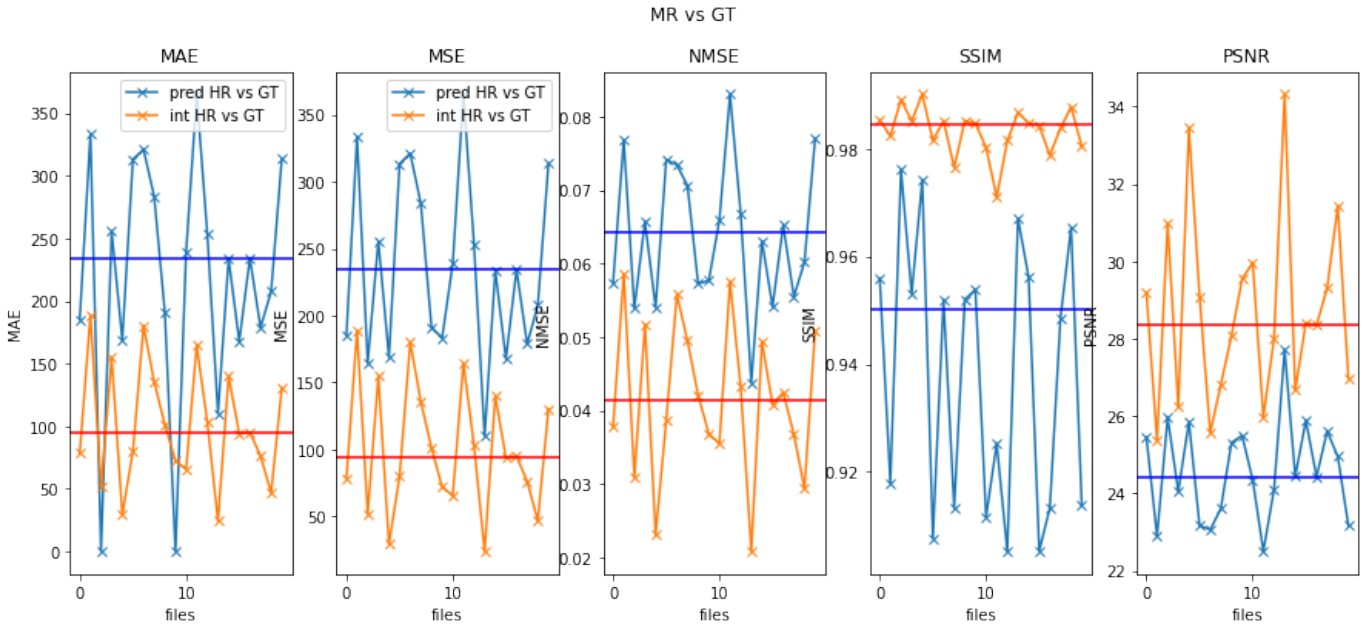


(d) Phase 2b

Fig. 12: Training and validation loss for simulation 9. Phase 1a and 2a are trained with a frozen encoder and phase 1b and 2b with unfrozen encoder weights. Details of simulation 9 can be found in Table I

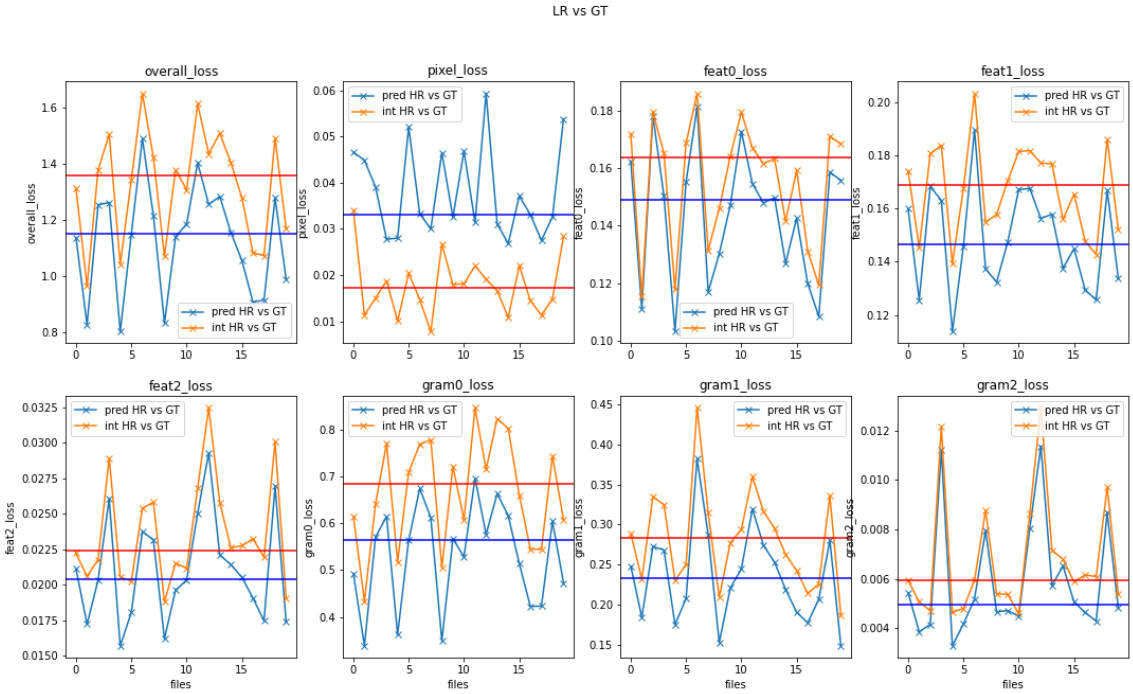


(a) Prediction and interpolation from LR test images.

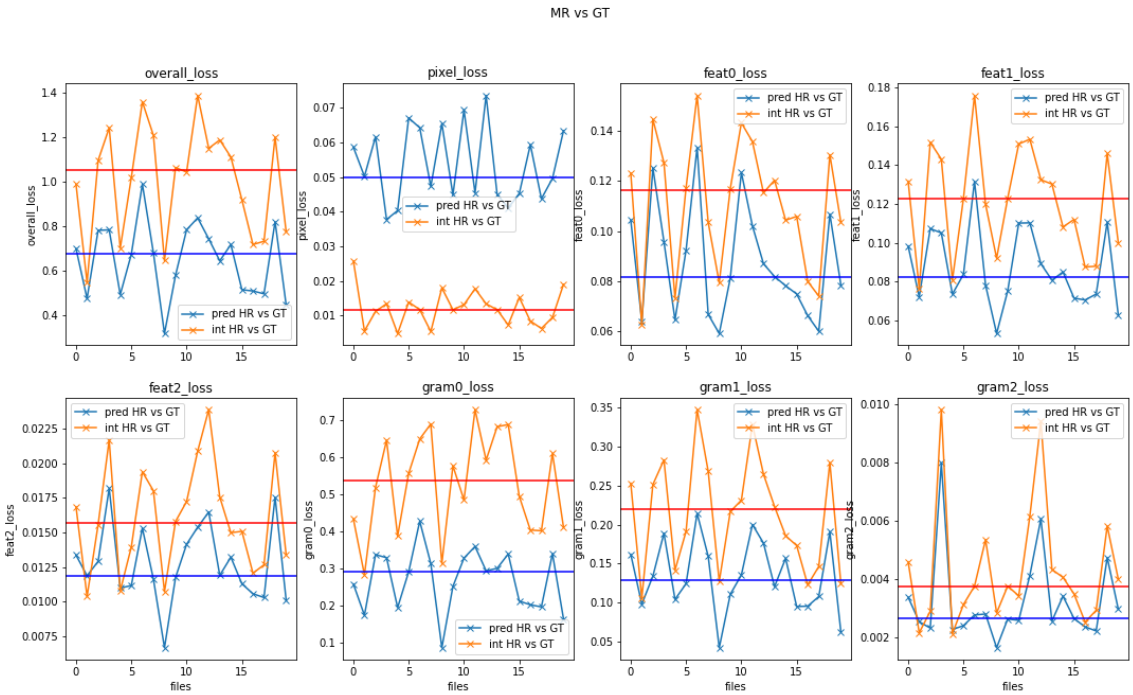


(b) Prediction and interpolation from MR test images.

Fig. 13: Different losses between HR predictions (from LR or MR input) and ground truth (GT) HR and losses between bicubic interpolated HR (from LR or MR) and ground truth HR. MAE, MSE, NMSE, SSIM and PSNR are evaluated for 20 test images. Median of values are shown in red for interpolated HR and in blue for predictions.



(a) Prediction and interpolation from LR test images.



(b) Prediction and interpolation from MR test images.

Fig. 14: Training losses between HR predictions (from LR or MR input) and ground truth (GT) HR and losses between bicubic interpolated HR (from LR or MR) and ground truth HR. Three feature losses, gram matrix losses, pixel loss and overall loss (sum of aforementioned losses) are evaluated for 20 test images. Median of values are shown in red for interpolated HR and in blue for predictions.

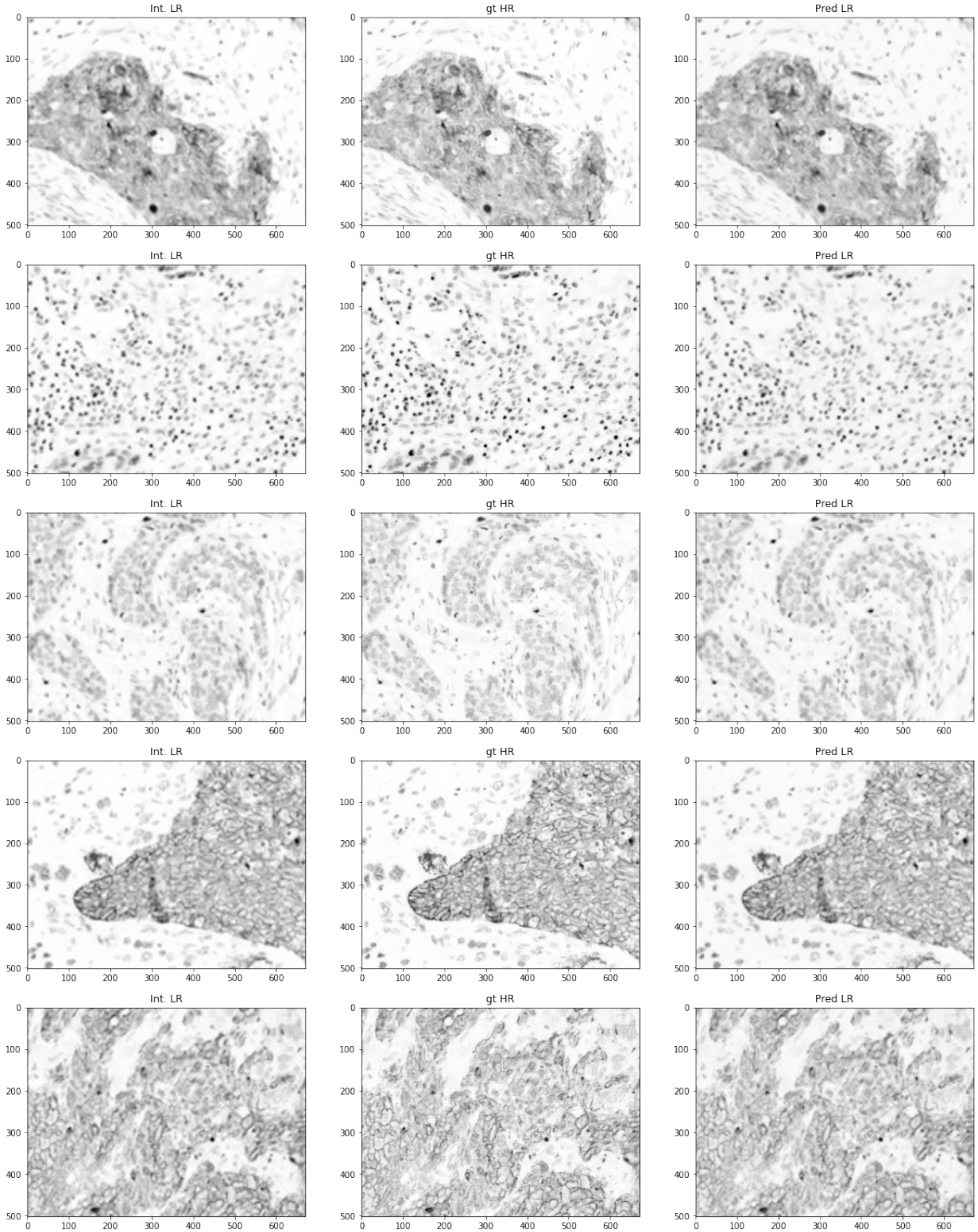


Fig. 15: HR predictions by our model from LR test data (right) compared to ground truth HR (middle) and the HR created by bicubic interpolation (left). Only one channel is shown here as each channel is a phenotype. Images are of size (3, 502, 672).

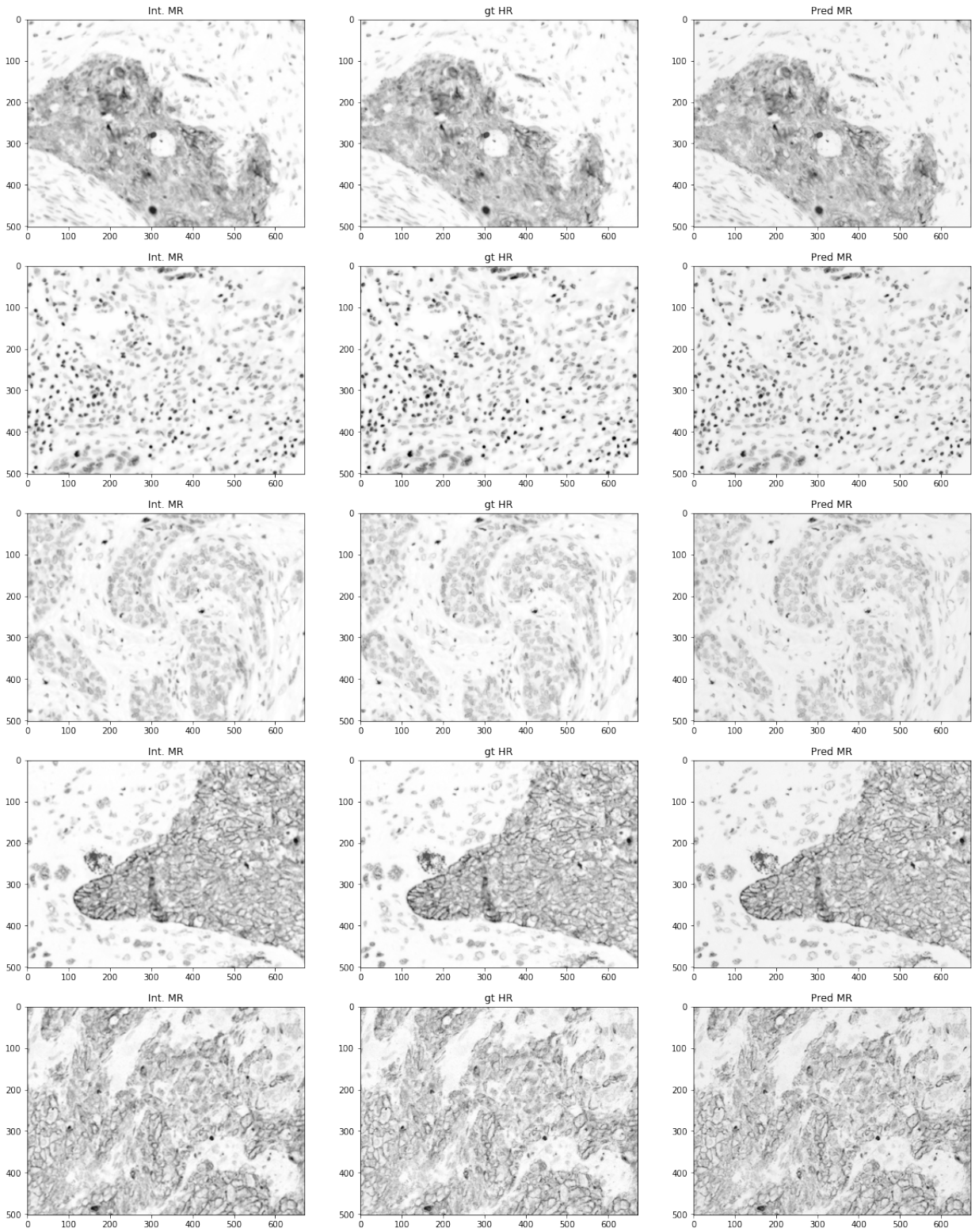


Fig. 16: HR predictions from MR test data (right) compared to ground truth HR (middle) and the HR created by bicubic interpolation (left). Only one channel is shown here as each channel is a phenotype. Images are of size (3, 502, 672).

TABLE I

Model	LR	MR	HR	numb. train images	numb. val images	Normalization	Transformations	Image type
Sim 1	(3, 500, 669)	(3, 1004, 1344)	(3, 1004, 1344)	1388	154	Imagenet stats	transf	JPEG
Sim 2	(3, 250, 334)	(3, 502, 672)	(3, 502, 672)	5552	616	Imagenet stats	transf	JPEG
Sim 3	(3, 250, 334)	(3, 502, 672)	(3, 502, 672)	5552	616	Imagenet stats	transf	JPEG
Sim 4	(3, 250, 334)	(3, 502, 672)	(3, 502, 672)	5552	616	Image stats	transf	JPEG
Sim 5	(3, 250, 334)	(3, 502, 672)	(3, 502, 672)	1851	205	Image stats	transf	tiff
Sim 6	(3, 250, 334)	(3, 502, 672)	(3, 502, 672)	1851	205	Image stats	none	tiff
Sim 7	(3, 125, 167)	(3, 502, 672)	(3, 502, 672)	1851	205	Image stats	none	tiff
Sim 8	(3, 125, 167)	(3, 250, 334)	(3, 502, 672)	1851	205	Imagenet stats	transf	tiff

Table I. Different simulations run for training our model. **Normalisation**: *Imagenet stats* when the data was normalised according to Imagenet statistics, e.g. range of [0, 1] and normalised using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]. *Image stats* means that they were normalised to their own statistics. **Transformations**: when not none, images were transformed according to the transformations described in section III-C1.

TABLE II

Model	x	y	Batch size	Num epochs	Weight decay	learning rate	Train loss	Val loss
Simul 2	(3, 500, 500)	(3, 500, 500)	4	10	1e-3	0.01	0.5513	0.5490
Sim 3	(3, 250, 250)	(3, 250, 250)	20	10	1e-3	0.01	0.6505	0.6240
Sim 4	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	0.01	0.6056	0.5875
Sim 5	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	0.01	0.6043	0.5753
Sim 6	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	0.01	0.4498	0.5015
Sim 7	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	0.01	0.3511	0.3460
Sim 8	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	0.01	0.6199	0.6041
Sim 9	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	0.01	0.5916	0.5254

Table II. Phase 1a. **Validation and training loss**: value at the end of the last epoch. **x and y**: input and output given to the model during training.

TABLE III

Model	x	y	Batch size	Num epochs	Weight decay	learning rate	Train loss	Val loss
Sim 2	(3, 500, 500)	(3, 500, 500)	4	10	1e-3	slice(1e-5, 1e-3)	0.5476	0.5245
Sim 3	(3, 250, 250)	(3, 250, 250)	20	10	1e-3	slice(1e-5, 1e-3)	0.6066	0.5747
Sim 4	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	slice(1e-5, 1e-3)	0.5945	0.5712
Sim 5	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	slice(1e-5, 1e-3)	0.5854	0.5634
Sim 6	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	slice(1e-5, 1e-3)	0.4250	0.5098
Sim 7	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	slice(1e-5, 1e-3)	0.3417	0.3417
Sim 8	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	slice(1e-5, 1e-3)	0.5808	0.5766
Sim 9	(3, 250, 334)	(3, 250, 334)	15	10	1e-3	slice(1e-5, 1e-3)	0.5410	0.4945

Table III. Phase 1b. **Validation and training loss**: value at the end of the last epoch. **x and y**: input and output given to the model during training.

TABLE IV

Model	x	y	Batch size	Num epochs	Weight decay	learning rate	Train loss	Val loss
Sim 2	(3, 1004, 1004)	(3, 1004, 1004)	1	10	1e-3	1e-3	0.6276	0.6512
Sim 3	(3, 502, 502)	(3, 502, 502)	5	10	1e-3	1e-3	0.7236	0.7507
Sim 4	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	1e-3	0.7106	0.7462
Sim 5	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	1e-3	0.7023	0.7366
Sim 6	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	1e-3	0.6394	0.8479
Sim 7	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	1e-3	0.6892	0.6805
Sim 8	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	1e-3	0.8395	0.8304
Sim 9	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	1e-3	0.7139	0.7763

Table IV. Phase 2a. **Validation and training loss**: value at the end of the last epoch. **x and y**: input and output given to the model during training.

TABLE V

Model	x	y	Batch size	Num epochs	Weight decay	learning rate	Train loss	Val loss
Sim 2	(3, 1004, 1004)	(3, 1004, 1004)	5	10	1e-3	slice(1e-6, 1e-4)	0.628	0.6488
Sim 3	(3, 502, 502)	(3, 502, 502)	5	10	1e-3	slice(1e-6, 1e-4)	?	?
Sim 4	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	slice(1e-6, 1e-4)	0.6948	0.7219
Sim 5	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	slice(1e-6, 1e-4)	0.685 3	0.713
Sim 6	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	slice(1e-6, 1e-4)	0.6141	0.7928
Sim 7	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	slice(1e-6, 1e-4)	0.6769	0.6657
Sim 8	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	slice(1e-6, 1e-4)	0.8301	0.8178
Sim 9	(3, 502, 672)	(3, 502, 672)	4	10	1e-3	slice(1e-6, 1e-4)	0.5578	0.5201

Table V. Phase 2b. **Validation and training loss**: value at the end of the last epoch. **x and y**: input and output given to the model during training.